SE - TP3 : Supermarché

4 novembre 2014

1 But du TP

Lors de ce TP vous allez simuler le fonctionnement d'un supermarché, de ses clients et de ses deux employés. La supérette dispose de quatre produits en stock, et les clients disposent d'une liste de courses indiquant la quantité souhaitée de chacun des quatre produits.

La figure 1 présente un schéma de la supérette, et sa composition exacte est décrite ci-dessous (constituants puis acteurs intervenants dans la simulation). La supérette est constituée de :

1. **un entrepôt** qui contient les stocks des différents produits avant leur mise en rayon. On pourra considérer que ces stocks sont illimités ;
2. **quatre rayons** qui contiennent les exemplaires disponibles à la vente de chacun des produits (*Sucre, Farine, Beurre, Lait*). Le nombre d'exemplaires d'un produit dans le rayon est limité par la taille du rayon.
3. **une rangée de chariots** dans laquelle chaque client doit prendre un chariot avant d'entrer dans le magasin. Les chariots sont diposés les uns dans les autres, et il n'y a qu'une file de chariots.
4. **une caisse** qui dispose d'un tapis où un client dépose ses produits un par un. Il ne peut y avoir qu'un seul client à un moment donné, qui dépose des marchandises sur le tapis. Dans un but de simplification, on ne prendra en considération que le dépôt sur le tapis par le client, et la prise des marchandises par l'employé. Le dépôt et la prise des produits seront faite dans le même ordre *(Le tapis est FIFO)*. Le tapis a une taille finie et est circulaire. On pourra le modéliser sous la forme d'une suite de cases pouvant contenir un produit ou le marqueur "client suivant".

Les intervenants à simuler sont :

1. **un chef de rayon** dont le rôle est de faire en sorte que les rayons soient le plus plein possibles. Il peut transporter jusqu'à 5 exemplaires de chaque produit à la fois, et passe son temps à aller à l'entrepôt, puis à faire le tour des rayons pour disposer le plus possible d'exemplaires de chaque produit dans le bon rayon. Le chef de rayon met les temps suivants pour ses opérations : 500 ms pour "faire le plein" à l'entrepôt, 200 ms pour se déplacer entre chaque rayon, ainsi qu'entre l'entrepôt et le premier rayon.
2. **un employé de caisse** dont le rôle est d'attendre qu'un client arrive à la caisse, et lorsqu'un client commence à déposer des éléments sur le tapis, de les prendre. Il s'arrête lorsque le tapis est vide. On simulera seulement la prise des éléments sur le tapis, jusqu'à trouver la marque "client suivant". Il faudra faire en sorte que le client et l'employé se synchronisent pour

effectuer le paiement. Ensuite, l'employé attend qu'un nouveau client dépose des marchandises sur le tapis ;

3. **Des clients** dont le rôle est d'effectuer la suite d'action suivante :
   (a) Décider d'une liste de courses ; [1]
   (b) Prendre un chariot dans la file. Si celle-ci est vide, attendre qu'un chariot y soit déposé ;
   (c) Parcourir les quatre rayons et prendre les exemplaires nécessaires. Un client ne peut quitter un rayon que lorsqu'il a pu prendre le nombre d'exemplaires fixé dans sa liste. Un client marche pendant 300 ms entre chaque rayon, les autres temps sont négligés.
   (d) Passer en caisse : s'il y a déjà un client qui passe ses objets en caisse, le client doit attendre que celui-ci ait fini pour commencer à placer ses marchandises. Le passage des clients en caisse peut ne pas être FIFO.
   (e) Placer ses produits sur le tapis. Le client met 20 ms à placer un élément sur le tapis. Une fois que tous ses produits ont été déposés, il place le marqueur de client suivant.
   (f) Attendre que l'employé ait fini de passer ses marchandises en caisse, afin d'effectuer le règlement.
   (g) Remettre son chariot dans la file.

   Les clients démarrent tous en même temps, et la simulation s'arrête lorsqu'il n'y a plus de clients à simuler.

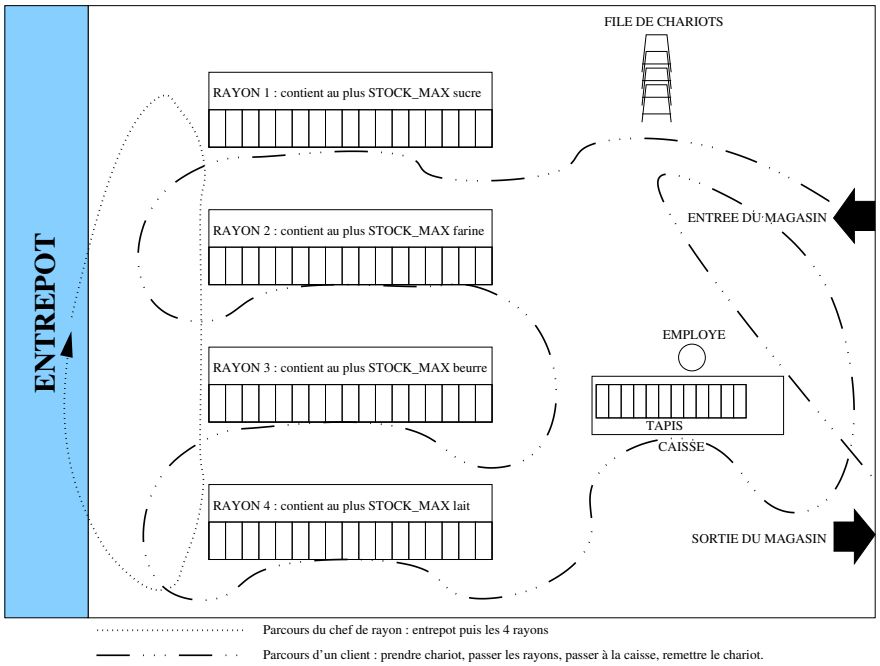

FIGURE 1 – Schéma de la supérette, avec le parcours d'un client et du chef de rayon.

1. Vous pourrez la générer aléatoirement

2 Structure de l'application

La classe `Supermarche.java` contiendra la méthode `main()` qui va créer l'ensemble des ressources et des processus. Des constantes importantes y seront définies :

- `Supermarche.NB_ELEM_PAR_CHGT` : nombre d'exemplaires de chaque article que le chef de rayon peut transporter dans sa tournée de remplissage des rayons ;
- `Supermarche.RAYON_STOCK_INIT` : le stock initial présent dans les rayons à l'ouverture du magasin ;
- `Supermarche.RAYON_STOCK_MAX` : nombre maximum d'exemplaires d'un produit dans un rayon ;
- `Supermarche.TPS*` : les temps, en ms, des diverses opérations décrites dans le sujet pour lesquelles le temps n'est pas négligé ;
- `Supermarche.TAILLE_TAPIS` : nombre maximum d'objets présents sur le tapis de caisse ;
- `Supermarche.NB_CHARIOTS` : nombre de chariots dans la file à l'ouverture du magasin.

Les clients seront représentés par des instances d'une classe `Client.java`. Tous les objets sont créés dans le `main()` de `Supermarche.java`, et pour accéder à un élément du supermarché, ces objets seront des attibuts de classe accessibles publiquement.

3 Instructions et conseils

Les affichages des différentes actions (par exemple "Client 2 prend un chariot" ou "Employé remet 3 Lait dans le rayon 4") doivent contenir explicitement le processus qui fait l'action, ainsi que sur quel objet partagé l'action est faite. Lorsque vous mettez un thread en attente sur un objet partagé, indiquez le dans les sorties, par exemple : "Client 4 ne peut plus prendre de Sucre, mise en attente sur Rayon 1".

La caisse Le problème de synchronisation le plus difficile se situe au niveau de la caisse. Voici quelques remarques :

- Il est judicieux d'utiliser un buffer de `Supermarche.TAILLE_TAPIS` places, avec gestion circulaore des indices.
- Le contenu d'une case sera soit un chiffre de 0 à 3 pour représenter un produit par son numéro, soit un marqueur spécifique représentant la marque "client suivant" (par exemple -1).
- Plus généralement, pensez avant de coder : déterminez d'abord quels sont les différents motifs d'exclusion, ceci vous aidera pour proposer une solution correcte (voire efficace - il n'est pas forcément judicieux de faire toute la synchronisation dans un seul moniteur.

4 Compte-rendu

Pour ce TP, vous devez rendre pour le 28 novembre 2014 23h59 CET (0,5 point sera enlevé par 3h de retard), par mail adressé à votre encadrant de TP et dont l'objet sera *[TP-SE] Groupe Nom1 Nom2* :

- les sources Java commentées (et correctement indentées), ainsi que l'information nécessaire à la compilation et l'exécution de votre projet ;
- des traces d'exécution avec 5 clients ;
- un compte rendu (au format **PDF**) qui répond aux spécifications suivantes :
  1. Il ne fera pas plus de 8 pages ;

2. Il contiendra votre décomposition en classes, en particulier quels objets sont des Threads, et quels objets sont partagés. Il n'est pas demandé de diagramme UML ;
3. La liste des motifs d'exclusion, les problèmes de synchronisation rencontrés, et comment vous les aurez résolus. Il pourra être judicieux de rapprocher les problèmes rencontrés de ceux rencontrés précédemment. Cette partie sera le cœur du rapport.
4. Toute information qui vous semble pertinente pour faciliter la compréhension de votre solution.

Le barême provisoire suivant est donné à titre indicatif :
– 8 points pour le compte rendu ;
– 12 points pour le programme et les traces d'exécution (fonctionnement correct, commentaires, lisibilité et pertinence des traces proposées).